Reflective Report

Abstract

This report analyses the development process of a small web application project in an agile team. The effects of using Scrum to manage the team are first described and the main development process of the project is outlined. This is followed by a review and reflection on some of the issues that arose during the actual operation of the team. These include a review of the Scrum process, a review of the level of compliance with the Scrum guidelines and a review of the performance of individuals on the team. Finally, there is a reflection on version control management of the project. The focus throughout the report is on the effectiveness and importance of the Scrum model for the management of agile teams.

Introduction

Our team has been using the Scrum Guide during the development of our client projects. The Scrum Guide describes the definition of Scrum, a lightweight framework that helps people, teams and organisations create value by providing adaptive solutions to complex problems [11]. Teams use the Scrum Guide to manage their teams and collaborate with each other and team members to achieve effective project operations. At the end of the project, the relevant functionality required by the client was already largely complete. We were involved in the creation of a fitness web application that was primarily used for communication and teaching between trainers and gym goers. Reflection and continuous process improvement is a fundamental part of agile software development. Reflecting on the issues encountered during the development process of this project can be of obvious help in future development. The development work of the project has been completed and this article will focus on my personal practice and reflection on the Scrum process and related development processes during this development process.

Reflect on the team's effectiveness in managing the project using Scrum as a distrusted team.

Because of COVID-19, most teams now work remotely, i.e., in distributed teams. Distance issues have the potential to lead to communication problems, poor information flow and members not being aware of each other's progress and problems [1]. Traditional teamwork also has problems with unequal information between team members, a lack of understanding of the importance of the project and a shortage of process awareness, where the various processes of the project are not connected smoothly enough.

When these situations arise, if left unattended, they can lead to slow projects, piles of tasks, information failures and less efficient communication. This can be fatal for a development team that needs to be efficient. Especially deadly for distributed teams. Scrum, one of the more popular agile development methodologies in the Internet industry, is currently able to solve these problems.

Scrum is an agile software development management methodology for iterative incremental software development processes. Agile is a word used to describe a process model concept which is different from the existing process model concepts [2]. Scrum is a process skeleton that includes a set of practices and predefined roles. Scrum was developed by Jeff Sutherland in 1993 and its goal is to become a development and management methodology that follows the principles of Agile methodology [3].

In Role Scrum, it is divided into 3 parts [4]:

- Scrum Master, the Scrum coach and team lead, who ensures that the team runs Scrum rationally and helps the team to remove barriers to implementation.
- The Product Owner, who sets the direction and vision for the product, defines the content, priorities and delivery times for the product release, and is responsible for the product's return on investment.
- The development team, a small cross-functional team under 10 people, with a team that has the range of skills needed to deliver usable software.

Managing a team through Scrum allows for clearer project objectives and clear timelines, facilitating the development process. Tasks are refined and goals are broken down, which helps the whole project to be achieved and increases team efficiency. Daily meetings and development plans are known to ensure the flow of information and efficient team communication. Review and retrospective meetings help to summarise the team's shortcomings, correct mistakes and continuously improve the work [5].

The results of using Scrum to manage projects are remarkable and impressive. As a member of the team, in my opinion, I think I did a good job in my role as Developer. I was able to create plans for the Sprint and predict the feasibility of each iteration and break down the User Story into more granular tasks of design, prototyping, coding, testing, etc., with detailed estimates of the workload. Make commits and merge with features made by other members at the specified time. Quickly fixes problems with team members if they arise. Communicates and collaborates well with team members and resolves disagreements promptly. Attends meetings on time and is a positive speaker at meetings. Conducts Daily Scrums on time. What I think needs to be improved in the future in this project is the lack of timely adjustments to the Christmas holiday schedule in response to external changes.

Difference between The Scrum Guide and our team's practice.

As a distributed group, and because of the impact of COVID-19, we can only work remotely. Scrum is very suitable for our group and can greatly improve our productivity.

First of all, we run our team according to the Scrum Guide, which has three pillars of opportunity empiricism, namely Transparency, Inspection and Adaptation. In terms of Transparency, the emerging processes and work of our team are visible to all and can be well inspected. If Transparency is low, it can lead to decisions that reduce value and increase risk. The team members know exactly what each other has done, what method or tool was used to implement a particular function, how it should be connected and how to call each other's methods. Members of the group also regularly inspections each other's code and leave comments on each other's code. This is a great way of identifying any potential discrepancies or problems. Regular inspections also allow us to catch bugs in the program that we could easily overlook on our own, the later a bug is caught, the more time it costs, and it is exponentially more expensive. This reduces the time cost of fixing bugs at the end of the future and greatly improves development efficiency. For Adaptation, all members of the group are supremely empowered on and the group members are also self-managed. This allows adjustments to be implemented as quickly as possible to minimise further deviations.

All roles in the team are focused on one goal, the Product Goal. Each member of the team creates their own value in each Sprint. Team members are also self-managed so that decisions can be made internally about who does what, when and how, which is very flexible. Our teams

are small enough to remain flexible, but big enough to get important work done in one Sprint. Smaller teams are better for communication and more efficient. Our teams can think in terms of agile development, which increases productivity and reduces uninspiring day-to-day labour.

It is worth mentioning that the scrum master in our team is on a rotating basis. In order for everyone to experience it once, it is set up so that one person can be the scrum master for one day in a week. This is not the same as the Scrum Guide. In my opinion, although this gives us a deeper understanding of the Scrum Master, it can only exist during the teaching period and the Scrum Master is very important in the team and is responsible for building the Scrum according to the Scrum Guide. Rotating the Scrum Master can lead to a fragmentation of responsibilities, which is not conducive to the proper execution of duties. Work can also increase in complexity when handing over.

Sprint is at the heart of Scrum, where ideas are converted into value. They are events of fixed duration to maintain their consistency. Our team has set the Sprint cycle to one week. Each Sprint is set according to the specific project requirements and the feedback provided by the client during the Sprint Review meeting. We also adhere to a strict guideline that no changes are made during the Sprint that would jeopardise the Sprint Goal. It is worth noting that when the length of a Sprint is stretched too far, the Sprint Goal may fail and complexity may rise, along with the risk. This is because if the Sprint is too long, the more features and methods the members develop, the more conflicts will be created. Each Sprint should be a short cycle.

In practice, our group's Sprint 2 lasted too long, from before the Christmas break to a week after it. This was very counterproductive to the iterative nature of the project. Although the set sprint goals achieved, and the lack of communication during the holiday period led to increased complexity in code interfacing and increased risk in merging.

A Sprint Planning meeting is held prior to the start of the iteration where team members select high priority product features from the feature list and incorporate them into the iteration, these features are then refined into User Stories and finally we each pair up and pledge to the User Story to form the Sprint task list. The team will have a daily meeting with only three topics: "What has been done since the last Daily Scrum Meeting?", "What problems are encountered during the work?", And "What will be done for the next scrum? "[6]. The burn-out chart is updated after the daily brief meeting. When each task in the Sprint is completed, a Sprint review meeting is held to present the Sprint results to the client, get a pass mark, record the defects and rework them to a usable version. The Sprint Retrospect meeting follows the Sprint Review meeting, where we reflect on the people, relationships, processes, tools, design, code and other dimensions of the Sprint process, and select and develop improvement plans. Through these meetings, group members have a clear idea of their own process, the team's overall process and what they should do next. Each step is made clear in its entirety. Even though the team works remotely, we are as productive as if we were working in one office every day.

Reflect on the team's effectiveness in using a workflow that uses branches and merge requests to manage source code control.

In the development of our projects, we use Git for version control. Git is a distributed version control system for keeping track of changes to any set of files, originally designed to coordinate work between programmers collaborating on source code during the software development process. Its goals include speed, data integrity, and support for distributed, non-

linear workflows (running thousands of parallel branches on different systems). GitLab is used as a software hosting platform. GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration and deployment pipeline features.

Just as code needs to be standardised, code management needs a clear process and standardisation. Git flow has a clear and controlled approach to branching and releasing. the advantage of Git flow is that it is clear and controlled, but the disadvantage is that it is relatively complex and requires the maintenance of two long-term branches (master and development).

Problems solved:

1. make the naming of existing branches more standardised.
2. make the role of each branch more clear
3. reduce code merge conflicts and code overwrites that can lead to bugs in production environments

In the early stages of development, after Client meetings, project requirements and briefs were received. We then met to discuss and create a series of User stories and created the group's master branch, which is an online stable version of the code that is not allowed to be committed. A dev branch was created in accordance with the Git versioning process and specification. dev is the branch where the code is developed, and all development is done in the dev branch. During the development process, we take branches from the dev branch for development when we receive a task assigned to us. When development is complete, we will ask to merge into the dev branch. If we encounter bugs or other issues, we pull the hotfix branch from the dev branch and fix them. Once the changes are made, they are merged back into the development branch. Using this workflow will give us fewer conflicts and facilitate the merging of code.

The use of feature branching and merging has greatly improved the efficiency of the team's work. Everyone knows exactly what function a branch is intended to achieve. Everyone knows how to manage the whole project. The team significantly improves efficiency by using git workflows during the development of projects.

From my point of view, I think I play the role of Developer well. There is a structured approach to branch creation. For the naming of branches, it is very clear that they are named according to their functional characteristics. Every time a merge request is submitted it is checked by a teammate and after a review. Actively working with team members on Hotfixes. Pulling branch information during development to ensure you get the latest news. In addition, I will process the information on the Label in time to put the completed user story on ready to test. After the merge is complete, the user story is closed. It helps to keep an overview of the overall project completion. In the preliminary work, the branch of development is not linked to the specific user story. This is not conducive to development. However, the branch was linked to its corresponding user story in later development after being pointed out in the review meeting.

Conclusion

Based on the concept of agile software development, this report applies the Scrum model and analyses various problems and reflections in the development of web applications in the

context of actual development situations. It is argued that reflection is an effective way to stabilise new agile projects, promote continuous improvement in development, and resolve team and project conflicts.

Risk Registers

Legal Risks

The increasing relevance of software systems in all economic and social sectors means that the legal implications relating to the development, procurement and use of software systems are becoming increasingly important. Legal issues are therefore not only important for the final product, but also need to be taken into account in every activity of the software development life cycle [7]. Mainly follows the GDPR regulations [8].

Risk	Description	Likelihood	Impact	Countermeasure
Data Protection	When software systems are used to process personal data, this can lead to problems such as the collection of user data by the software.	High	Medium	Limited collection of user details and excludes the collection of sensitive information. Ensuring that user information is only available to trainers when authorised by the user. Protect individual's right to privacy with respect to processing personal data.
Intellectual Property	Regulation on the allocation of property rights on software. The software may involve issues such as software intellectual property rights.	Low	Low	The software is original to the group and the essential functions are based on client requirements. Copyright issues are protected by law.
User Rights.	Regulations and laws protect the rights of consumers. The question of whether the rights of users can be guaranteed and whether they can be given sufficient rights is one that the software needs to face.	High	Medium	Users have the right to make their own choices. The user's rights are implemented in accordance with the law.
Contracts and agreements	Regulation on the commercial transactions between developer and client/costumer. The software may be a lack of user agreements.	Medium	Medium	Addition of relevant documents such as a legally compliant user agreement, which is provided to the customer to read at the time of registration and gives the user the right to choose.

Ethical Risks

The ethical issues involved are many and varied, however, it is helpful to focus on just four. These may be summarized by means of an acronym – **PAPA** [9].

Risk	Description	Likelihood	Impact	Countermeasure
Privacy	Individuals have the right to decide whether to disclose information about themselves or their associations.	High	High	The personal information function in the software allows the user to fill in only basic personal information or to choose to keep it confidential.
Accuracy	Who is responsible for the truthfulness as well as the accuracy of the information in the software? And who will be responsible for compensation.	Medium	Medium	Identification of those responsible, and mechanisms for compensation. It is the software's responsibility to ensure that all fitness information is correct and to mark and warn of dangerous fitness activities.
Property	Ownership of intellectual property rights in the software.	Low	Medium	The ownership of the software is specified within the contract. If this is not stipulated in the contract, the software belongs to the developer by default.
Access	The software party's rights of access to information and under what conditions and with what safeguards.	High	Low	The software party is responsible for protecting user data. Provides data storage as well as preservation. Safeguarding user data from leakage.

Social Risks

Risk	Description	Likelihood	Impact	Countermeasure
Software Accessibility	The software should also be improved in terms of accessibility.	High	Medium	Add accessibility to make the software work for anyone. For example, add on-screen reading and subtitles.
Financial Issues	The cost of software development and the economic impact it has on society. The software may bring some economic impaction to the fitness industry.	Low	Low	Budgeting for the production and distribution of the software. Evaluate the economic impact of the software on society and plan accordingly.
Competition Issues	The same type of software creates competition in the community. This software may not be conducive to competition.	Low	Medium	Improve your core competencies and do market research. Add features where appropriate to attract users.

Professional Risks

As professionals, they are expected to abide by the rules and professional standards set by the British Computer Society. Follow its guidance on the conduct of members in professional matters [10].

Risk	Description	Likelihood	Impact	Countermeasure
Professional Competence and Integrity	Software developers may not be aware of their knowledge base or may not have an accurate idea of their level. This can lead to later developments that do not meet the requirements.	High	Low	Continually improve the professional knowledge, skills and abilities of developers and maintain developers' awareness of relevant technical developments, procedures and standards in their field.
Duty to Relevant Authority	A conflict-of-interest situation may arise between the developer and the relevant authorities or the software developed may be in breach of relevant licensing or legislative requirements.	Low	Low	Avoid conflicts as far as possible and read the relevant legislation and policies in detail. Comply with the regulations of the relevant authorities, exercise professional responsibility as a developer with due care and exercise professional judgement accordingly.

Security Risks

Risk	Description	Likelihood	Impact	Countermeasure
Software Stability	The stability of the software refers to the error rate and performance degradation trend of the software under certain stress conditions during a runtime period. The stability of the application server, database server and other systems within the environment in which it operates is also observed.	High	Low	Edge testing is required to verify the stability of the software. Software stability emphasises the stability of the software architecture from a software development point of view, meaning that changes to requirements, code etc. have as little impact as possible on the software system, which is the primary task to be addressed by the architectural design.
Software Compatibility	Software compatibility is an important indicator of how good software is and refers to the ability to transfer software from one environment to another.	High	Medium	Test the software for compatibility with different operating systems/platforms, different browsers and different types of databases before release. Fix incompatibilities that arise.
Software troubleshooting	When the software is too complex, its error correction capability becomes problematic.	Medium	Medium	Increase the number of staff running and maintaining the software to deal with problems as they are identified.

Reference:

[1]. Fiore S M, Salas E, Cuevas H M, et al. Distributed coordination space: toward a theory of distributed team process and performance[J]. Theoretical Issues in Ergonomics Science, 2003

[2]. Martin, R. C., "Agile software development: principles, patterns, and practices",Upper Saddle River, N.J., Pearson Education, 2003

[3]. Pham, A., et al., "Scrum in action Agile software project management and development". Boston, Mass., Course Technology PTR, 2011

[4]. Woodward, E., et al., A practical guide to distributed Scrum. Upper Saddle River, NJ, IBM Press, 2010

[5]. Derby E, Larsen D, Schwaber K. Agile retrospectives: Making good teams great[M]. Pragmatic Bookshelf, 2006

[6]. Falls, M. , Books24x7 Inc., "Inside the minds the software business : how top companies design, develop & sell successful products & applications", Inside the minds. Boston, Mass., Aspatore, 2004

[7]. Rejas-Muslera R J, Urbán M A S, April A. Legal Risk Management Process in Software Projects: An Action Research Study[J].

[8]. Information Commission Office, Guide to the General Data Protection Regulation (GDPR), 2018

[9]. Mason R O. Four ethical issues of the information age[J]. MIS quarterly, 1986: 5-12.

[10].White S. COMP1205 LEPP: BCS Code of Conduct[J]. 2015.

[11]. Ken, S.; Jeff, S.; The Scrum Guide ; November 2020.